EK287384616US

# BATCH SUBMISSION API

## FIELD OF THE INVENTION

The present invention relates to an apparatus and method for reducing remote API network traffic and increasing API performance through the use of a batch submission API.

## BACKGROUND OF THE INVENTION

When backing up and restoring directory limits in an operating system, add, set and get routines work only on a single directory path. Since the operation on a single path may require multiple Add, Set or Get Application Program Interface (API) calls, restoration of a large amount of directory paths can generate a large amount of network traffic. Moreover, restoration of a large amount of directory paths can require sending and receiving the API and the data for each operation across the network. For example, in a network of computers, a first computer in the network may have a failed hard drive and a second computer may contain backup data for the first computer. An administrator may want to invoke a backup and restore routine. If there happened to be 64,000 target directories and all 64,000 directories had to be updated, there would be 128,000 round trip operations across the network. In some cases the backup and restore operations could take several days. What is needed is a way to reduce the time required to complete the backup and restore operations. In order to reduce the traffic, one way would be to batch all of the operations at one computer and send the batch to the second computer.

Batch processing has been used to increase the efficiency of data handling. United States Patent 5,889,896 discloses batch processing of raw image data which has been scanned into the memory of a processing system. System throughput is enhanced by permitting a workstation to execute a task on a batch of node index file records and to return the modified index file records to the stage of a work queue. However, the prior art does not address the problem of applying batch submission to reduce network traffic when backing up and restoring directory limits in an operating system.

## SUMMARY OF THE INVENTION

The invention which meets the needs identified above is an apparatus and method aggregating API calls within a higher level Batch Submission API (BSA) having a sender BSA (SBSA) and a receiver BSA (RBSA). Specifically, one or more directory limit records needed for the restoration process on a particular server are grouped by the SBSA into one large buffer. The SBSA validates batch records prior to transmitting them to a receiver. The SBSA does not transmit the records if either (1) all of the records fail pre-submission validation or (2) a certain percentage of the records fail pre-submission validation.

Optionally, the SBSA may transmit records marked invalid which will not be processed if the count of invalid records is less than a predetermined number. A record fails if flags are not set or paths are not covered. For example, a pass rate for records may be set at 80%. If 80% of the records in the batch pass pre-submission validation, the SBSA will transmit the records to the second computer. The remaining 20% of the records are marked invalid. Optionally, some invalid records may be sent. If the number of records marked invalid exceeds the pre-determined number, the valid records are not sent. If the number of records marked invalid does not exceed the pre-determined number, the valid records are sent.

Once the SBSA has performed the pre-submission validation of the records, a buffer and count (BC) is transmitted to RBSA at a second computer. The BC contains records passing pre-submission validation checks and a count of the records contained within the buffer. The count of the records contained in the buffer is equal to the total count of records in the backup file minus the number of records that fail pre-submission validation (records failing pre-submission validation are excluded). The RBSA installed on the second computer is responsive to a BC and would allow the BC to be transmitted to the second computer. The RBSA would ignore individual or multiple API calls and would only accept a BC. The number of network transmissions would thereby be reduced. The RBSA processes each of the directory limits records contained in the BC at the second computer. If one or more of the records failed, a vector of return code structures containing the index or ordinal matching the failing record along with the return code would be returned and the count of failed records contained in the vector. If all records in the BC completed successfully, then no return code structures would be returned and the count of failed records would be zero. By having the RBSA process the BC at the second computer, the back and forth traffic that would occur if each operation were performed individually across the network is significantly reduced. Moreover, system efficiency is enhanced because the needed API calls to process the record are issued locally at the second computer. The RBSA would handle the logic to Add a directory limit where one did not exist, Set a directory limit if one existed and Get information if required for an Add or Set.

## BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 depicts a distributed data processing system in which the invention may be implemented;

Figure 2 depicts a computer in which the software to implement the invention may be stored;

Figure 3 depicts a flow chart for the batch submission API;

Figure 4 depicts a flow chart for the server process.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Figure 1 depicts a pictorial representation of a distributed data processing system in which the present invention may be implemented and is intended as an example, and not as an architectural limitation, for the processes of the present invention. Distributed data processing system **100** is a network of computers which contains a network **102**, which is the medium used to provide communications links between various devices and computers connected together within distributed data processing system **100**. Network **102** may include permanent connections, such as wire or fiber optic cables, or temporary connections made through telephone connections, personal computers or network computers. Distributed data processing system **100** may include additional servers, clients, and other devices not shown. In the depicted example, distributed data processing system **100** is the Internet with network **102** representing a worldwide collection of networks and gateways that use the TCP/IP suite of protocols to communicate with one another. Distributed data processing system **100** may also be implemented as a number of different types of networks, such as for example, an intranet, a local area network (LAN), or a wide area network (WAN).

Figure 2 depicts computer **200**. Although the depicted embodiment involves a personal computer, a preferred embodiment of the present invention may be implemented in other types of data processing systems. An exemplary hardware arrangement for computer **200** follows. Keyboard **222** and display **223** are connected to system bus **210**. Read only memory (ROM) **230** contains, typically, boot strap routines and a Basic Input/Output System (BIOS) utilized to initialize Central Processing Unit (CPU) **220** at start up. Random Access Memory (RAM) **240** represents the main memory utilized for processing data. Drive controller **250** interfaces one or more disk type drives such as floppy disk drive **252**, CD ROM **254** and hard disk drive **256**. The number and type of drives utilized with a particular system will vary depending upon user requirements. A network interface **260** permits communications to be sent and received from a network. Communications port **270** may be utilized for a dial up connection to one or more networks while network interface **260** is a dedicated interface to a particular network. Programs for controlling the apparatus shown in Fig. 2 are typically stored on a disk drive and then loaded into RAM for execution during the start-up of the computer.

Figure 3 depicts a flow chart of the Sender Batch Submission Application Program Interface (SBSA). The SBSA is located in a first computer containing back up files which are to be used to restore directory limits to the operating system of a second computer that has had a failure. In the preferred embodiment, the first computer is a client computer and the second computer is a server computer. The program is invoked (**302**) and initialized (**304**). "X" represents the total count of records in the backup files. "Y" represents the number of records that fail validation. At initialization, "X" is set to zero and "Y" is set to zero. The acceptable threshold "Z" is set (**306**). The program then determines whether there is a record to read (**308**). If there is a record to read, then the program will read the next record (**310**). Next, X is set equal to X + 1 (**312**). The program then performs record validation (**314**). Next, the program determines if a validation error has occurred (**316**). A validation error occurs if flags are not set or paths are not covered. If a validation error has occurred, then Y is set equal to Y + 1 (**318**) and the record is stored in a failed buffer (**320**). After the record is stored in the failed buffer, the program goes to step **308** to determine if there is a record to read. If a validation error has not occurred, then the record is stored in a batch buffer (**322**) and the program returns to step **308** to determine if there is another record to read.

At step **308**, if the program determines that there is not another record to read (the last record has been examined), then the program goes to step **324** to determine whether Y is greater than 0. If Y is greater than 0, then the program goes to step **326** and determines whether Y/X is less than or equal to "Z." "Z" is a variable that represents the acceptable error threshold percentage for pre-submission validation. The value of "Z" is given to the SBSA; the program supplying the batch records. For example, if Z is set to 20%, then if Y/X is less than or equal to 20%, the program would send the records in the batch buffer and a count of the records in the batch buffer. The batch and the count of records shall be referred to as the batch and count (BC). The count of records is equal to the total count of records minus the number of records that fail pre-submission validation (X -Y) because failed records are not sent. If Y/X is not less than or equal to Z (greater than Z), then a report is displayed that error threshold "Z" has been exceeded (**328**). If the acceptable error threshold "Z" is exceeded, the batch as a whole is considered in error and not submitted. If Y/X is less than or equal to "Z" then the batch of records minus the failed records and count of records (BC) is sent (**332**).

If at step **308**, the program determines that Y equals 0 (Y is not greater than 0), then the BC is sent to the receiver (**330**). When a BC is sent, a report will be received from the second computer as described below.

Figure 4, depicts the process at the second computer. The second computer contains the Receiver Batch Submission Application Program Interface (RBSA). The RBSA ignores individual or multiple API calls and only accepts a BC. If a BC is sent, the RBSA receives the BC (**404**) and determines whether there is a record to be read (**406**). If there is a record to be read, the RBSA reads the next record (**408**). The RBSA then performs an operation on the first record contained in the BC (**410**). If the record fails (**412**), the RBSA generates an error record (**414**) and goes to step **406**. The RBSA then determines whether there is another record to be read (**406**). If there is another record to be examined, the RBSA reads the next record (**408**). If there is not another record to be read, the RBSA determines whether there are failed records (**416**) . If there are failed records, a vector of return code structures containing the index or ordinal matching the failing record along with the return code is returned along with the count of failed records contained in the vector (**418**). If there are no failed records, a report is sent. If all records in the BC completed successfully, then no return code structures would be returned and the count of failed records would be zero. By having the RBSA process the BC at the second computer, the back and forth traffic that would occur if each operation were performed individually across the network is significantly reduced. Moreover, system efficiency is enhanced because the needed API calls to process the records in the BC are issued locally at the second computer. The RBSA would handle the logic to Add a directory limit where one did not exist, Set a directory limit if one existed and Get information if required for an Add or Set. In the event that the second computer does not have a RBSA installed, the BC will not be recognized and an error code will be returned to the first computer. Upon receipt of the error code, the first computer will process the records without using the SBSA, or in other words, as the first computer would have done without having the enhancement of the SBSA.

The advantages provided by the present invention should be apparent in light of the detailed description provided above. The description of the present invention has been presented for purposes of illustration and description, but is not limited to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in order to best explain the principles of the invention the practical application and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.